

УДК 519.87

ЛОКАЛЬНЫЙ ПОИСК С ОКРЕСТНОСТЬЮ ЭКСПОНЕНЦИАЛЬНОЙ МОЩНОСТИ ДЛЯ ЗАДАЧИ БАЛАНСИРОВКИ НАГРУЗКИ НА СЕРВЕРЫ *)

И. А. Давыдов, П. А. Кононова, Ю. А. Кочетов

Аннотация. Для решения задачи балансировки нагрузки на серверы предложен метод локального поиска с оригинальной окрестностью экспоненциальной мощности. Исследуются варианты локального поиска с рандомизированными версиями такой окрестности. Приводятся результаты численных экспериментов, свидетельствующие о высокой эффективности предложенного подхода.

Ключевые слова: локальный поиск, задача о назначениях, балансировка нагрузки.

Введение

Рассмотрим следующую задачу балансировки нагрузки на серверы [4]. Имеется набор серверов, на которых размещены диски (точнее, их образы). На дисках хранятся интернет сайты с разнородной информацией. Пользователи посещают сайты, что создаёт определённую нагрузку на серверы. Эта нагрузка меняется со временем и характеризуется несколькими параметрами (например, нагрузка процессора, требуемая оперативная память и др.). Для каждого диска известна активность пользователей в течение планового периода. Эта активность позволяет определить нагрузку серверов в каждый момент времени по каждому параметру. Время предполагается дискретным, т. е. измерение нагрузки производится, например, каждую минуту или секунду. Если нагрузка по каждому параметру не превосходит заданного порога, то сервер находится в рабочем режиме. В противном случае сервер работает с перегрузкой. Чтобы избежать перегрузки, диски можно перемещать с одного сервера на другой. Такое перемещение требует определённых вычислительных затрат. Будем называть их накладными расходами. Предполагается, что

*) Исследование выполнено в Новосибирском госуниверситете при финансовой поддержке Минобрнауки РФ (договор № 02.G25.31.0054).

для каждого диска известны накладные расходы по каждому параметру при его изъятии с сервера и подключении к любому другому серверу. Начальное распределение дисков по серверам считается известным. Задача состоит в том, чтобы перераспределить диски по серверам и достичь минимальной суммарной перегрузки на всём плановом периоде при ограничениях на накладные расходы по каждому серверу.

Для решения этой задачи в [4] предложен приближённый алгоритм с апостериорной оценкой точности. Задача представляется в терминах частично-целочисленного линейного программирования. Целочисленные переменные заменяются непрерывными, и оптимальное решение соответствующей задачи линейного программирования используется для получения приближённого решения и оценки его погрешности. Более точно, линейное программирование позволяет зафиксировать часть дисков на серверах, сократить размерность задачи и решить оставшуюся подзадачу точно методом ветвей и границ (пакет CPLEX). Такой подход позволяет быстро находить оптимальное решение на примерах с нулевой перегрузкой и даёт хорошие результаты при большой перегрузке серверов. При малой перегрузке линейное программирование фактически не даёт целочисленных положительных компонент, размерность задачи не сокращается и алгоритм не работает.

В настоящей работе предлагается другой подход, основанный на идеях локального поиска. Наряду с естественными окрестностями малой мощности (сдвинуть диск на другой сервер или поменять его на другой диск) рассматривается новая оригинальная окрестность экспоненциальной мощности. На каждом сервере выбирается по одному диску, и производится перераспределение этих дисков между серверами. Каждый сервер снова получает по одному диску, а суммарная перегрузка серверов минимизируется. Решение задачи о назначениях даёт наилучшее перераспределение дисков. Исследуются различные способы выбора дисков для перераспределения и эффективность локального поиска с такой окрестностью.

1. Математическая модель

Введём следующие обозначения: S — множество серверов, D — множество дисков, T — плановый период, R — множество характеристик нагрузки (CPU, RAM, ...), c_{drt} — нагрузка диска d в момент времени t по характеристике r , \bar{c}_{sr} — пороговая нагрузка сервера s по характеристике r , x_{ds}^0 — начальное распределение дисков по серверам, $b_{sdr}^w(b_{sdr}^e)$ — накладные расходы по характеристике r на вставку (изъятие) диска d

в сервер s , $B_{sr}^w(B_{sr}^e)$ — предельно допустимые накладные расходы по характеристике r на вставку (изъятие) дисков в сервер s .

Переменные задачи: $x_{ds} = 1$, если диск d ставится на сервер s , $x_{ds} = 0$ в противном случае, y_{str} — перегрузка на сервере s в момент времени t по характеристике r .

С использованием введённых обозначений задача балансировки нагрузки может быть записана в виде задачи частично-целочисленного линейного программирования [4]:

$$\min \sum_{s \in S} \sum_{t \in T} \sum_{r \in R} y_{str} \quad (1)$$

при ограничениях

$$y_{str} \geq \sum_{d \in D} c_{drt} x_{ds} - \bar{c}_{sr}, \quad s \in S, t \in T, r \in R, \quad (2)$$

$$\sum_{s \in S} x_{ds} = 1, \quad d \in D, \quad (3)$$

$$\sum_{d \in D} b_{sdr}^w x_{ds} (1 - x_{ds}^0) \leq B_{sr}^w, \quad s \in S, r \in R, \quad (4)$$

$$\sum_{d \in D} b_{sdr}^e x_{ds}^0 (1 - x_{ds}) \leq B_{sr}^e, \quad s \in S, r \in R, \quad (5)$$

$$y_{str} \geq 0, x_{ds} \in \{0, 1\}, \quad d \in D, s \in S, t \in T, r \in R. \quad (6)$$

Целевая функция (1) задаёт суммарную перегрузку серверов на всём плановом периоде. Неравенство (2) позволяет определить эту перегрузку для каждого сервера в каждый момент времени по каждой характеристике. Равенство (3) гарантирует, что каждый диск будет помещён в точности на один сервер. Неравенства (4) и (5) ограничивают накладные расходы по каждому серверу и характеристике при изъятии и вставке дисков.

Представление задачи в терминах целочисленного линейного программирования позволяет использовать для её решения коммерческое программное обеспечение, например, IBM ILOG CPLEX, GUROBI, AMPL и др. К сожалению, наличие большого разрыва целочисленности при переходе к непрерывным переменным x_{ds} не даёт возможности найти точное решение задачи на примерах даже средней размерности. Например, при $|S| = 20$, $|D| = 200$, $|T| = 150$, $|R| = 2$ расчёты могут занимать более суток без гарантии найти точное решение. Отчасти это объясняется тем, что задача NP-трудна в сильном смысле даже при $|T| = |R| = 1$ и нулевых накладных расходах.

Действительно, известная задача о 3-разбиении сводится к этому частному случаю задачи (1)–(6). Напомним, что в задаче о 3-разбиении заданы $3m$ чисел $a_i, i = 1, \dots, 3m$, и граница B такая, что $B/4 < a_i < B/2$, $\sum_{i=1}^{3m} a_i = mB$. Можно ли так разбить эти числа на m непересекающихся подмножеств, чтобы сумма в каждом подмножестве была равна B ? Эта задача NP-трудна в сильном смысле. Для её сведения к указанному частному случаю задачи (1)–(6) достаточно положить $|S| = m, |D| = 3m, \bar{c}_{sr} = B$ и $c_{drt} = a_d$ для всех $s \in S, d \in D$.

Данный факт подталкивает к разработке приближённых методов решения задачи. Одним из приоритетных направлений в этой области является разработка методов локального поиска, которые хорошо зарекомендовали себя при решении многих NP-трудных задач дискретной оптимизации [14].

2. Окрестности

При разработке методов локального поиска центральное место занимает вопрос выбора окрестностей, определяющих понятие соседства на множестве допустимых решений задачи. От того, какие окрестности будут использоваться в локальном поиске, существенно зависит результат работы алгоритма. Окрестности малой мощности позволяют быстро находить наилучшее соседнее решение, но часто приводят к локальным оптимумам с большим отклонением от глобального оптимума. Окрестности большой мощности, как правило, дают меньшую погрешность, но требуют больших вычислительных затрат на каждом шаге локального поиска. На сегодняшний день нет и, возможно, никогда не появится единого правила выбора окрестности. Для каждой задачи эту проблему приходится решать заново, учитывая специфику целевой функции и ограничений. Более того, для каждой задачи, по-видимому, можно предложить несколько окрестностей разной мощности и, как следствие, с разными множествами локальных оптимумов. Чередование этих окрестностей в ходе локального поиска часто приводит к высокоэффективным методам [3, 7, 10]. Ниже будут предложены три окрестности, две из которых имеют полиномиальную мощность от числа серверов и дисков. Третья имеет экспоненциальную мощность. Её элементами являются оптимальные решения вспомогательной задачи о назначениях. Исследование окрестностей большой мощности представляет несомненный интерес [8, 9, 15] и позволяет повысить эффективность численных методов [1, 6, 11, 12].

Пусть пара $x = (x_{ds})$, $y = (y_{str})$ — допустимое решение задачи (1)–(6). Заметим, что переменные y однозначно определяются по переменным x . Поэтому в дальнейшем допустимое решение будем обозначать через x . Пусть $\text{Move}(x)$ — множество допустимых решений, которые могут быть получены из x путём выбора одного диска и перемещением его на другой сервер. Мощность такой окрестности можно оценить сверху величиной $|D| \cdot |S|$.

Вторая окрестность, которую обозначим через $\text{Swap}(x)$, будет содержать все допустимые решения, получаемые из x выбором двух дисков на разных серверах и заменой их друг на друга. Мощность такой окрестности можно оценить сверху величиной $|D|^2$.

Несмотря на полиномиальную мощность этих окрестностей в алгоритме будут использоваться их рандомизированные версии. Такой приём повышает эффективность поиска и сокращает время каждой итерации [2, 5]. Под рандомизированной окрестностью $\text{Move}_q(x)$ с параметром $0 < q < 1$ будем понимать часть окрестности $\text{Move}(x)$, в которую каждый элемент входит с вероятностью q независимо от других элементов. Аналогично определяется окрестность $\text{Swap}_q(x)$, при этом значение параметра q может быть другим.

Рассмотрим теперь новую окрестность $\text{Assign}(x)$ экспоненциальной мощности, которая строится следующим образом. На каждом сервере сформируем множество дисков D_s , $s \in S$, извлечение которых с данного сервера не нарушает ограничений (5). По некоторому правилу выберем из каждого множества по одному диску. Множество выбранных дисков обозначим через \bar{D} . Если множество D_s оказалось пустым для некоторого сервера s , то будем считать, что с этого сервера был извлечён фиктивный диск с нулевой нагрузкой по каждому параметру и нулевыми накладными расходами на изъятие и вставку его на любой сервер. Правила выбора дисков для непустых множеств D_s будут обсуждаться позже.

Обозначим через Δ_{ds} суммарную перегрузку сервера s после изъятия с него одного диска и вставке диска d из множества \bar{D} :

$$\Delta_{ds} = \max \left\{ 0, \sum_{t \in T} \sum_{r \in R} \sum_{d' \in D \setminus \bar{D}} c_{d'rt} x_{d's} - \bar{c}_{sr} + c_{drt} \right\},$$

если такая вставка не нарушает ограничений (4), в противном случае полагаем $\Delta_{ds} = +\infty$.

Введём новые переменные

$$z_{ds} = \begin{cases} 1, & \text{если диск } d \text{ из } \bar{D} \text{ ставится на сервер } s, \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда наилучшее распределение дисков по серверам можно получить из решения следующей задачи о назначениях:

$$\begin{aligned} \min \sum_{d \in \overline{D}} \sum_{s \in S} \Delta_{ds} z_{ds}, \\ \sum_{d \in \overline{D}} z_{ds} = 1, \quad s \in S, \\ \sum_{s \in S} z_{ds} = 1, \quad d \in \overline{D}, \\ z_{ds} \in \{0, 1\}, \quad d \in \overline{D}, s \in S. \end{aligned}$$

Целевая функция этой задачи задаёт суммарную перегрузку серверов на всём плановом периоде по всем параметрам. Первое ограничение требует установки ровно одного диска на каждый сервер. Второе ограничение гарантирует, что каждый диск будет установлен только на один сервер.

Заметим, что если множества D_s , $s \in S$, не пусты, то решение задачи о назначениях не меняет число дисков на каждом сервере. Следующий приём позволяет избавиться от этого недостатка. Будем считать, что на каждом сервере имеется некоторый фиктивный диск с нулевой нагрузкой по каждому параметру в каждый момент времени и с нулевыми накладными расходами на изъятие и вставку. Включение такого диска в множество \overline{D} фактически означает, что с некоторого сервера не было изъято никакого диска. Следовательно, в оптимальном решении задачи о назначениях какой-то сервер получит фиктивный диск, что может приводить к уменьшению числа дисков на единицу.

Как отмечалось выше, элементами данной окрестности являются оптимальные решения задачи о назначениях. Значит, мощность окрестности равна произведению чисел $|D_s| + 1$, $s \in S$, если в задаче о назначениях только одно оптимальное решение. В противном случае мощность окрестности ещё больше. Поэтому снова будем применять идею рандомизации окрестности. Выбор элемента из множества D_s будем проводить случайным образом с некоторым распределением, зависящим (или не зависящим) от суммарной перегрузки сервера. Ниже приведены результаты исследований для различных правил выбора дисков, но, забегая вперед, можно сказать, что равномерное распределение на множестве элементов D_s часто приводит к хорошим результатам.

В заключение раздела обратим внимание на следующее обстоятельство. Пользуясь равномерным распределением, получаем некоторую задачу о назначениях, т. е. случайный элемент окрестности $\text{Assign}(x)$. Если этот элемент совпадает с x , т. е. диски вернулись на исходные серверы, то это ещё не означает, что получен локальный оптимум. Окрестность $\text{Assign}(x)$ не содержит решений с большей перегрузкой, чем у решения x . Найти же элемент окрестности с наименьшей перегрузкой представляется нетривиальной задачей. Поэтому в ходе локального поиска будут порождаться наудачу некоторые элементы этой окрестности, и лучшее решение будет предъявляться в качестве результата такого поиска.

3. Алгоритм локального поиска

Предлагаемый ниже алгоритм локального поиска (Multi-start Local Search (MLS)) [13] начинается свою работу с начального решения $x^0 = (x_{ds}^0)$ и последовательно применяет к нему две процедуры локального поиска Local Search по окрестностям Move_{q_1} и Swar_{q_2} , а затем процедуру локального улучшения Local Improve по окрестности Assign . На каждом шаге процедуры Local Search осуществляется переход от текущего решения к наилучшему соседнему решению сначала по окрестности Move_{q_1} до получения локального оптимума, а затем по окрестности Swar_{q_2} до тех пор, пока не будет получен локальный оптимум x' по объединению этих окрестностей. Далее к этому решению x' применяется процедура Local Improve, в ходе которой несколько раз решается задача о назначениях со случайным выбором дисков с каждого сервера. К полученному решению x'' снова применяется процедура Local Search и Local Improve с целью улучшить текущее решение. Такие попытки повторяются до тех пор, пока i_{\max} раз не удаётся улучшить текущее решение. Затем вычисления повторяются снова со стартового решения x^0 . Наилучшее найденное решение x^* предъявляется в качестве ответа.

АЛГОРИТМ MLS (q_1, q_2, i_{\max})

ШАГ 1. Положить $x \leftarrow x^0, x^* \leftarrow x^0, i \leftarrow 0$, выбрать значения параметров i_{\max}, q_1, q_2 .

ШАГ 2. While $i \leq i_{\max}$ do:

ШАГ 2.1. $x' \leftarrow \text{Local Search}(x, q_1, q_2)$.

ШАГ 2.2. $x'' \leftarrow \text{Local Improve}(x')$.

ШАГ 2.3. Если $x'' = x$, то $i \leftarrow i + 1$, иначе $x \leftarrow x'', i \leftarrow 0$.

ШАГ 3. Если решение x лучше рекорда x^* , то $x^* \leftarrow x$.

ШАГ 4. Если не выполнен критерий остановки, то $x \leftarrow x^0$ и вернуться на шаг 2.

ШАГ 5. Предъявить рекорд x^* .

На шаге 4 в качестве критерия остановки используются либо число рестартов или достижение заданной границы по целевой функции, например, оптимального значения в соответствующей задаче линейного программирования, либо происходит остановка по времени счёта. Возвращение на шаг 2 применяется для диверсификации поиска. Рандомизация окрестностей позволяет получать всё новые и новые локальные оптимумы по окрестностям Move и Swap, а применение задачи о назначениях даёт возможность находить среди них решения с малой погрешностью.

4. Численные эксперименты

Представленный алгоритм локального поиска запрограммирован в среде Delphi 7.0 и протестирован на PC Intel Core i5 3.20GHz, RAM 4Gb 32-bit.

Экспериментальные исследования проводились на классе примеров из [4]: $|S| = 20$, $|D| = 200$, $|T| = 150$, $|R| = 2$. Величины x_{ds}^0 , задающие начальное распределение дисков по серверам, порождались случайным образом с равномерным распределением. Для каждого диска случайным образом выбирался один сервер. Для вычисления нагрузки и накладных расходов сначала каждому диску приписывалось значение средней нагрузки $c_d, d \in D$. Её выбирали из интервала $[50, 100]$ с равномерным распределением, а затем полагали

$$c_{drt} = c_d/10 + \mu_{drt}, \quad b_{sdr}^w = c_d/10 + \nu_{sdr}, \quad b_{sdr}^e = c_d/10 + \theta_{sdr},$$

$$s \in S, d \in D, r \in R,$$

где величины $\mu_{drt} \in [-20, 20]$, $\nu_{sdr}, \theta_{sdr} \in [-3, 3]$ также выбирались независимо с равномерным распределением из указанных интервалов.

Первый вычислительный эксперимент касался фиктивных дисков. Их применение позволяет менять число дисков на серверах, при этом логично использовать фиктивные диски на серверах с малой нагрузкой. В ходе экспериментов подсчитывалась суммарная нагрузка по каждому серверу и несколько (до пяти) наименее загруженных серверов получали фиктивные диски при формировании задачи о назначениях. Результаты тестирования показали, что применение фиктивных дисков фактически не сказывается на результатах поиска. По-видимому, это объясняется

тем, что на шаге 2.1 локального поиска используется окрестность Move, которая также меняет число дисков на серверах. Стартовое решение x' на шаге 2.2 уже оказывается достаточно сбалансированным, и фиктивные диски скорее «вредят» локальному поиску по окрестности Assign, исключая часть серверов. В дальнейших экспериментах фиктивные диски не используются, хотя их применение к начальному решению даёт положительный эффект.

Второй вычислительный эксперимент проводился для сравнения различных правил выбора дисков из множеств D_s , $s \in S$. Рассматривались следующие три правила:

- выбор диска с помощью равномерного распределения;
- серверы сортировались по суммарной нагрузке и делились на два равномоощных множества: лёгкие и тяжёлые, на каждом сервере выбиралось случайным образом несколько дисков и брался самый лёгкий для лёгкого сервера и самый тяжёлый в противном случае;
- серверы делились на три почти равномоощных подмножества: лёгкие, средние и тяжёлые, для средних использовалось первое правило выбора дисков, для остальных — второе.

Результаты тестирования не выявили явного преимущества одного из этих правил. Отдавать предпочтение лёгким дискам на лёгких серверах кажется оправданным, но большой плановый период $|T| = 150$ делает такие характеристики, как лёгкий-тяжёлый, слишком грубыми, а более точный анализ по времени представляется крайне затруднительным. В связи с этим далее будет применяться первое правило как наиболее простое.

В третьем эксперименте исследовалось расположение в допустимой области решений, получаемых на шаге 3 алгоритма. Заметим, что локальный поиск многократно стартует с одного и того же начального решения x^0 и применяет только процедуры монотонного улучшения. Рандомизация окрестностей должна способствовать получению различных решений, но они могут оказаться близки друг к другу, что крайне нежелательно. Являясь локальными оптимумами по рассматриваемым окрестностям, они скорее всего мало отличаются по целевой функции. В табл. 1 представлены попарные расстояния для 10 наилучших найденных решений. При $|D| = 200$ максимально возможное расстояние Хэмминга между решениями равно 400, а среднее значение в таблице не меньше 250. Никакие два из этих решений не находятся близко друг к другу. Следовательно, выбранные значения $q_1 = 0.2$, $q_2 = 0.15$ для рандомизации окрестностей Move и Swap позволяют находить разные по

структуре решения. На рис. 1 показаны значения целевой функции решений, получаемых на шаге 3. Эти значения заметно отличаются друг от друга. Можно сказать, что дополнительных процедур диверсификации поиска не требуется несмотря на то, что каждый раз поиск начинается снова с решения x^0 .

Т а б л и ц а 1

Расстояния Хэмминга для 10 наилучших решений

$i \backslash j$	1	2	3	4	5	6	7	8	9	10
1	0	268	280	282	270	278	264	252	276	278
2	268	0	264	288	284	274	270	254	270	272
3	280	264	0	278	288	262	262	264	286	264
4	282	288	278	0	250	288	262	272	278	264
5	270	284	288	250	0	276	256	262	282	272
6	278	274	262	288	276	0	278	284	286	286
7	264	270	262	262	256	278	0	258	266	274
8	252	254	264	272	262	284	258	0	262	272
9	276	270	286	278	282	286	266	262	0	274
10	278	272	264	264	272	286	274	272	274	0

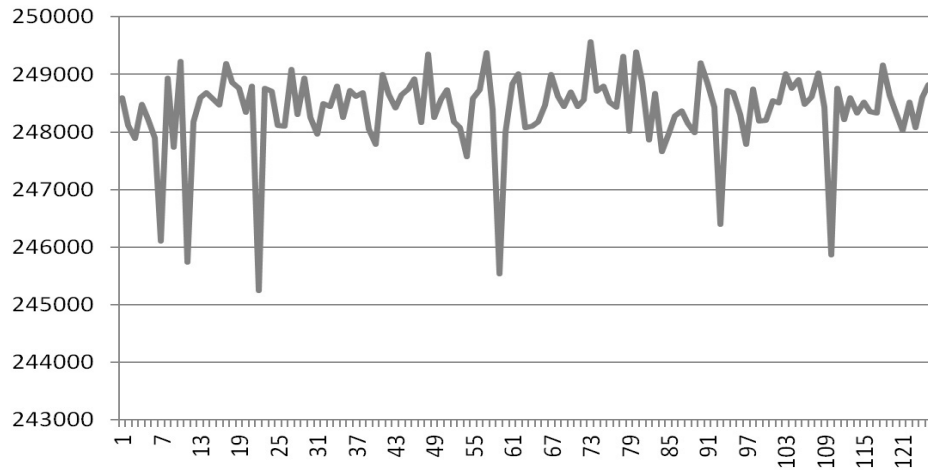


Рис. 1. Типичное поведение метода

В табл. 2 и 3 приведены результаты расчётов на примерах из работы [4]. В первом столбце указано значение пороговой загрузки \bar{c}_{sr} . В столбцах со второго по пятый приведены результаты расчётов из [4]. Столбцы 2 и 3 демонстрируют результаты, полученные при решении линейной релаксации задачи (1)–(6). В столбце t_{LP} указано время счёта

в секундах. Отметим, что значение целевой функции F_{LP} задачи линейного программирования даёт нижнюю оценку оптимума. В столбцах 4 и 5 представлены результаты приближённого алгоритма, основанного на последовательной фиксации дисков по решению задачи линейного программирования. В столбце t_{ILP} указано время счёта в секундах. Алгоритм принудительно останавливался через 50 минут. Значение целевой функции лучшего найденного решения приведено в столбце F_{ILP} . В некоторых примерах 50 минут работы оказалось недостаточно для получения практически значимых результатов.

Т а б л и ц а 2

Результаты расчётов при $B_{sr}^w = B_{sr}^e = 50$

\bar{c}_{sr}	t_{LP}	F_{LP}	t_{ILP}	F_{ILP}	t_{MLS}	F_{MLS}
850	2	0	6	0	1	0
840	2	0	7	0	1	0
830	2	0	17	0	1	0
820	2	0	70	0	2	0
810	2	0	3000	1834	6	0
800	2	0	3000	2480	14	0
750	198	0	—	—	1200	13965
700	423	96911	3000	188306	1200	129511
650	160	382399	607	407792	1200	383448
600	58	682399	85	702039	1	682399

Т а б л и ц а 3

Результаты расчётов при $B_{sr}^w = B_{sr}^e = 150$

\bar{c}_{sr}	t_{LP}	F_{LP}	t_{ILP}	F_{ILP}	t_{MLS}	F_{MLS}
850	2	0	8	0	1	0
840	2	0	21	0	1	0
830	2	0	11	0	1	0
820	2	0	3000	4	2	0
810	2	0	2100	0	2	0
800	2	0	3000	1370	3	0
750	200	0	—	—	1200	8424
700	734	87700	—	—	1200	120862
650	182	382399	3000	469294	1200	382560
600	62	682399	3000	761107	1	682399

Последние два столбца демонстрируют результаты работы алгоритма локального поиска. В столбце t_{MLS} указано время работы алгоритма

в секундах. В качестве критерия остановки использовалось либо ограничение по времени работы — не более 20 минут, либо достижение нижней оценки линейного программирования.

Отметим, что при высоких значениях пороговой нагрузки задача легко решается точно и алгоритм локального поиска находит оптимальное решение менее чем за секунду расчётов. Малые значения пороговой нагрузки также не представляют больших трудностей. При $\bar{c}_{sr} = 600$ значение нижней оценки достигается за секунду расчётов. Для примеров со значением пороговой нагрузки 800–820 удалось найти оптимальные решения, которые не были найдены в [4]. В примерах со значениями пороговой нагрузки 750–650 достичь нижней оценки не удалось. Однако в примерах со значениями $\bar{c}_{sr} = 650$ лучшие найденные решения отличаются от значения нижней оценки менее чем на 1%.

Согласно [4] наиболее трудные примеры соответствовали пороговому значению $\bar{c}_{sr} = 700$. В следующих экспериментах использовалось именно это значение пороговой нагрузки. Табл. 4 показывает результаты расчётов при $B_{sr}^w = B_{sr}^e = 50$ для 10 различных тестовых примеров из указанного класса. Сравнение трёх последних столбцов позволяет сделать вывод о преимуществе нового алгоритма, хотя для одного (третьего) примера результаты оказались хуже предшественника. Тем не менее последней столбец показывает, что относительное уклонение от нижней оценки $\varepsilon = 100\%(F_{MLS} - F_{LP})/F_{MLS}$ достаточно мало даже в этом случае.

Т а б л и ц а 4

Результаты расчётов при $\bar{c}_{sr} = 700$

Пример	F_{LP}	F_{ILP}	F_{MLS}	$\varepsilon(\%)$
1	234 387	331 819	246 780	5,0
2	327 380	367 169	329 868	0,8
3	486 154	498 036	531 918	8,6
4	404 395	488 143	417 768	3,2
5	346 045	393 182	353 489	2,1
6	341 400	475 071	384 536	11,2
7	407 088	428 234	407 594	0,1
8	256 666	325 084	281 827	8,9
9	322 371	361 414	325 038	0,8
10	549 565	567 503	561 315	2,1

5. Заключение

Для задачи балансировки нагрузки на серверы предложен метод локального поиска, в котором наряду с полиномиальными по мощности

окрестностями используется оригинальная окрестность экспоненциальной мощности. Элементами этой окрестности являются оптимальные решения задачи о назначениях. Она получается изъятием дисков по одному с каждого сервера и перераспределением их между серверами оптимальным образом. Экспериментальные исследования свидетельствуют о высокой эффективности локального поиска с такой окрестностью.

В качестве направлений дальнейших исследований интересно было бы найти наиболее трудные в вычислительном отношении примеры и исследовать на них поведение алгоритмов. Например, можно генерировать случайным образом примеры с заранее известным оптимумом, просто разбивая значения \bar{c}_{sr} на положительные слагаемые по одному для каждого диска на сервере. Аналогичная идея использовалась ранее для задачи упаковки в контейнеры. Полученные таким способом примеры оказались «крепким орешком» для многих численных методов.

ЛИТЕРАТУРА

1. Береснев В. Л., Гончаров Е. Н., Мельников А. А. Локальный поиск по обобщённой окрестности для задачи оптимизации псевдоболевых функций // Дискрет. анализ и исслед. операций. — 2011. — Т. 18, № 4. — С. 3–16.
Beresnev V. L., Goncharov E. N., Melnikov A. A. Local search over generalized neighborhood for an optimization problem of pseudo-Boolean functions // J. Appl. Industr. Math. — 2012. — Vol. 6, N 1. — P. 22–30.
2. Давыдов И. А., Кочетов Ю. А., Младенович Н., Уросевич Д. Быстрые метаэвристики для дискретной задачи о $(r|p)$ -центроиде // Автоматика и телемеханика. — 2014. — Т. 75, № 4. — С. 106–119.
Davydov I., Kochetov Yu., Mladenovic N., Urosevich D. Fast metaheuristics for the discrete $(r|p)$ -centroid problem // Autom. Remote Control. — 2014. — Vol. 75, N 4. — P. 677–687.
3. Кононова П. А., Кочетов Ю. А. Локальный поиск с чередующимися окрестностями для задачи Джонсона с пассивным буфером // Дискрет. анализ и исслед. операций. — 2012. — Т. 19, № 5. — С. 63–82.
Kononova P. A., Kochetov Yu. A. The variable neighborhood search for the two machine flow shop problem with a passive prefetch // J. Appl. Industr. Math. — 2013. — Vol. 7, N 1. — P. 54–67.
4. Кочетов Ю. А., Кочетова Н. А. Задача балансировки нагрузки на серверы // Вестн. Новосиб. гос. ун-та. Сер. Информ. технологии. — 2013. — Т. 11, вып. 4. — С. 71–76.
5. Мельников А. А. Рандомизированный локальный поиск для дискретной задачи конкурентного размещения предприятий // Автоматика и телемеханика. — 2014. — Т. 75, № 4. — С. 134–152.
Melnikov A. Randomized local search for the discrete competitive facility

- location problem // Autom. Remote Control. — 2014. — Vol. 75, N 4. — P. 700–714.
6. **Ahuja R. K., Ergun O., Orlin J. B., Punnen A. P.** A survey of very large-scale neighborhood search techniques // Discrete Appl. Math. — 2002. — Vol. 123, N 1–3. — P. 75–102.
 7. **Davydov I. A., Kochetov Yu. A., Carrizosa E.** A local search heuristic for the $(r|p)$ -centroid problem in the plane // Comput. Oper. Res. — 2014. — Vol. 52, Part B. — P. 334–340.
 8. **Gutin G.** Exponential neighborhood local search for the traveling salesman problem // Comput. Oper. Res. — 1999. — Vol. 26. — P. 313–320.
 9. **Gutin G., Yeo A.** Small diameter neighborhood graphs for the traveling salesman problem: four moves from tour to tour // Comput. Oper. Res. — 1999. — Vol. 26. — P. 321–327.
 10. **Hansen P., Mladenovich N.** Variable neighborhood search // Eur. J. Oper. Res. — 2001. — Vol. 13. — P. 449–467.
 11. **Kochetov Yu., Alekseeva E., Levanova T., Loresh M.** Large neighborhood local search for the p -median problem // Yugoslav. J. Oper. Res. — 2005. — Vol. 15, N 1. — P. 53–63.
 12. **Kochetov Yu., Kononova P., Paschenko M.** Formulation space search approach for the teacher/class timetabling problem // Yugoslav. J. Oper. Res. — 2008. — Vol. 18, N 1. — P. 1–11.
 13. **Marti R.** Multi-start methods // Handbook of metaheuristics. — Dordrecht: Kluwer Acad. Publ., 2003. — P. 355–368.
 14. **Talbi E.-G.** Metaheuristics: from design to implementation. — Berlin: Wiley, 2009. — 624 p.
 15. **Yannakakis M.** Computational complexity // Local search in combinatorial optimization. — Chichester: Wiley, 1997. — P. 19–55.

Давыдов Иван Александрович,
e-mail: vamn.davydov@gmail.com
Кононова Полина Александровна,
e-mail: polik83@ngs.ru
Кочетов Юрий Андреевич,
e-mail: jkochet@math.nsc.ru

Статья поступила
11 апреля 2014 г.
Переработанный вариант —
22 мая 2014 г.